JOMO KENYATTA UNIVERSITY OF AGRICULTURE AND TECHNOLOGY

SCHOOL OF COMPUTING & INFORMATION TECHNOLOGY

DEPARTMENT: IT AND COMPUTING

COPAS

DATA ANALYSIS PROJECT

PROJECT NAME: WEATHER DATA DIVE

BY

JOSHUA KYENGO KIOKO

SCT213-C002-0063/2021

ABSTRACT

This research project offers a thorough weather dataset that has been carefully selected to provide comprehensive weather analysis and forecasting. A wide range of climatic factors, including temperature, humidity, precipitation, wind speed, atmospheric pressure, and more, are included in the dataset and were gathered from trustworthy source that is National Centers for Environmental Information (NCEI).

The dataset's 1/1/2012 to 31/12/2012 temporal coverage range offers a wide historical viewpoint to examine changing weather patterns and trends.

ACKNOWLEDGEMENT

I want to express my gratitude to the department head, Mr. James Mbao, the professors, and the entire school administration for giving me the tools and expertise I needed to complete this specific assignment. Finally, I want to thank my guardians for providing me with the practical tool I need to become a dependable gearhead and be able to meet my demands.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

1.1 Background

Weather data is essential to many aspects of our everyday lives and businesses, including transportation, agriculture, and disaster preparedness. Understanding and effectively utilizing weather data can help individuals and organizations make informed decisions, improve safety, and optimize operations.

1.2 Objective

The goal of this project is to clean data from a dataset and then produce insightful visualizations using the cleansed data. The project's objectives are to highlight the value of data cleansing in the data analysis process and the effectiveness of visualization in extracting information that is useful from the data.

LITERATURE OVERVIEW

1.3 Overview of data analysis

In order to derive useful insights and make wise judgments, data analysis is a crucial process that comprises analyzing, cleaning, manipulating, and interpreting data. It is essential to many different industries, such as finance, healthcare, weather forecasting, and scientific research.

1.4 Methodological review

"Data science without a strong methodological foundation is akin to sailing a ship without navigation instruments – directionless and susceptible to wandering into the waters of uncertainty." One quote from Advances in Data Science and Analysis (2012) that encapsulates the idea that Data science is a multidisciplinary field that encompasses various methodologies and techniques to extract knowledge and insights from data. In order to solve complicated problems and arrive at wise judgments, it includes taking an organized approach to processing data and utilizing statistical techniques, machine learning algorithms, and domain knowledge.

Researchers and practitioners emphasize the significance of robust methods as data science continues to develop in order to ensure the accuracy and repeatability of outcomes.

CHAPTER II

PROCESSING OF DATA

2.1 Data Collection

Using the most convenient way of obtaining datasets for analysis which means that one has obtain uncleaned data formally from a company or industry which deals with a lot of data, I decided to go for the weather agency that archives weather data for historical purposes. I obtained my data from The National Centers for Environmental Information (NCEI), which offers the general public a variety of datasets—many of which are undoubtedly dirty—that I could analyze and present visually.

here is an overview of the data provided by the organization;

	A	B	C	D	E	F	G	H
1	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather
2	1/1/2012 0:00	-1.8	-3.9	86	4	8	101.24	Fog
3	1/1/2012 1:00	-1.8	-3.7	87	4	8	101.24	Fog
4	1/1/2012 2:00	-1.8	-3.4	89	7	4	101.26	Freezing Drizzle,Fog
5	1/1/2012 3:00	-1.5	-3.2	88	6	4	101.27	Freezing Drizzle,Fog
6	1/1/2012 4:00	-1.5	-3.3	88	7	4.8	101.23	Fog
7	1/1/2012 5:00	-1.4	-3.3	87	9	6.4	101.27	Fog
8	1/1/2012 6:00	-1.5	-3.1	89	7	6.4	101.29	Fog
9	1/1/2012 7:00	-1.4	-3.6	85	7	8	101.26	Fog
10	1/1/2012 8:00	-1.4	-3.6	85	9	8	101.23	Fog
11	1/1/2012 9:00	-1.3	-3.1	88	15	4	101.2	Fog
12	1/1/2012 10:00	-1	-2.3	91	9	1.2	101.15	Fog
13	1/1/2012 11:00	-0.5	-2.1	89	7	4	100.98	Fog
14	1/1/2012 12:00	-0.2	-2	88	9	4.8	100.79	Fog
15	1/1/2012 13:00	0.2	-1.7	87	13	4.8	100.58	Fog
16	1/1/2012 14:00	0.8	-1.1	87	20	4.8	100.31	Fog
17	1/1/2012 15:00	1.8	-0.4	85	22	6.4	100.07	Fog
18	1/1/2012 16:00	2.6	-0.2	82	13	12.9	99.93	Mostly Cloudy
19	1/1/2012 17:00	3	0	81	13	16.1	99.81	Cloudy
20	1/1/2012 18:00	3.8	1	82	15	12.9	99.74	Rain
21	1/1/2012 19:00	3.1	1.3	88	15	12.9	99.68	Rain
22	1/1/2012 20:00	3.2	1.3	87	19	25	99.5	Cloudy
23	1/1/2012 21:00	4	1.7	85	20	25	99.39	Cloudy
24	1/1/2012 22:00	4.4	1.9	84	24	19.3	99.32	Rain Showers
25	1/1/2012 23:00	5.3	2	79	30	25	99.31	Cloudy

Weather Data

Ready

Fig2.1 Data provided in excel

2.2 Data Cleaning

Data cleaning is the process of changing or eliminating garbage, incorrect, duplicate, corrupted, or incomplete data in a dataset. There are several techniques to clean data, however in this case we must import libraries, load data, handle missing data, format the data, deduplicate the data, and save the cleaned data.

```
In [11]:
#importing the module
import pandas as pd  # data processing, CSV file I/O
import numpy as np # linear algebra
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import constants
# Replace 'file_path.csv' with the actual file path of your CSV file
file_path = "C:\\Users\\JOSHUA\\OneDrive\\Desktop\\Tableau PJT\\weather dataset\\Weather Data.csv"
#Load Data
df = pd.read_csv(file_path)
#Initial Exploration
df.info()
df.shape

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8784 entries, 0 to 8783
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Date/Time         8784 non-null   object
 1   Temp_C            8784 non-null   float64
 2   Dew Point Temp_C  8784 non-null   float64
 3   Rel Hum_%         8784 non-null   int64
 4   Wind Speed_km/h   8784 non-null   int64
 5   Visibility_km     8784 non-null   float64
 6   Press_kPa         8784 non-null   float64
 7   Weather           8784 non-null   object
dtypes: float64(4), int64(2), object(2)
memory usage: 549.1+ KB

Out[11]: (8784, 8)
```

Fig2.2 Data uploaded to the anaconda package for cleaning

```
In [11]: df.describe()
Out[11]:
```

	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa
count	8784.000000	8784.000000	8784.000000	8784.000000	8784.000000	8784.000000
mean	8.798144	2.555294	67.431694	14.945469	27.664447	101.051623
std	11.687883	10.883072	16.918881	8.688696	12.622688	0.844005
min	-23.300000	-28.500000	18.000000	0.000000	0.200000	97.520000
25%	0.100000	-5.900000	56.000000	9.000000	24.100000	100.560000
50%	9.300000	3.300000	68.000000	13.000000	25.000000	101.070000
75%	18.800000	11.800000	81.000000	20.000000	25.000000	101.590000
max	33.000000	24.400000	100.000000	83.000000	48.300000	103.650000

```
In [26]:
#Handling Missing Data
df.dropna(inplace=True)

In [27]:
#Deduplication
df.drop_duplicates(inplace=True)

In [4]:
#to get the dimension of the data
df.shape
Out[4]: (8784, 8)

In [5]: df.index
```

Fig2.3 Checking data for missing values and duplicated values

2.3 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a critical step in the data analysis process, where analysts and data scientists examine the dataset to understand its structure, characteristics, and relationships between variables. The primary goal of EDA is to gain insights into the data, identify patterns, detect outliers, and develop a deeper understanding of the underlying trends.

```
In [6]:  column = df.columns
         for columns in column:
             print("->",columns)

         -> Date/Time
         -> Temp_C
         -> Dew Point Temp_C
         -> Rel Hum_%
         -> Wind Speed_km/h
         -> Visibility_km
         -> Press_kPa
         -> Weather

In [7]:  df.dtypes

Out[7]:  Date/Time           object
         Temp_C             float64
         Dew Point Temp_C   float64
         Rel Hum_%            int64
         Wind Speed_km/h      int64
         Visibility_km      float64
         Press_kPa          float64
         Weather             object
         dtype: object

In [8]:  #number of unique values
         df.nunique()

Out[8]:  Date/Time          8784
         Temp_C              533
         Dew Point Temp_C    489
         Rel Hum_%            83
         Wind Speed_km/h      34
         Visibility_km        24
```

Fig2.4 Understanding the structure of the data

```
In [9]:  wind_speed = df['Wind Speed_km/h'].nunique()
         print('Wind Speed_km/h:-',wind_speed)

         Wind Speed_km/h:- 34

In [10]: #find unique number of wind speeds
         df.head(2)

Out[10]:
```

	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather
0	1/1/2012 0:00	-1.8	-3.9	86	4	8.0	101.24	Fog
1	1/1/2012 1:00	-1.8	-3.7	87	4	8.0	101.24	Fog

```
In [11]: df['Wind Speed_km/h'].unique()

Out[11]: array([ 4,  7,  6,  9, 15, 13, 20, 22, 19, 24, 30, 35, 39, 32, 33, 26, 44,
                43, 48, 37, 28, 17, 11,  0, 83, 70, 57, 46, 41, 52, 50, 63, 54,  2],
               dtype=int64)

In [18]: df['Weather'].value_counts()

Out[18]: Mainly Clear                              2106
         Mostly Cloudy                             2069
         Cloudy                                    1728
         Clear                                     1326
         Snow                                       390
         Rain                                       306
         Rain Showers                               188
         Fog                                        150
         Rain,Fog                                   116
         Drizzle,Fog                                 80
         Snow Showers                                60
```

Fig2.5 Data snippet of insights

```
In [19]: # variance of the Humidity
         df.rename(columns = {'Rel Hum_%': 'Humidity'}, inplace = True)
```

```
In [25]: df[df['Wind Speed_km/h'] == 4]
Out[25]:
```

	Date/Time	Temp_C	Dew Point Temp_C	Humidity	Wind Speed_km/h	Visibility_km	Press_kPa	Weather
0	1/1/2012 0:00	-1.8	-3.9	86	4	8.0	101.24	Fog
1	1/1/2012 1:00	-1.8	-3.7	87	4	8.0	101.24	Fog
96	1/5/2012 0:00	-8.8	-11.7	79	4	9.7	100.32	Snow
101	1/5/2012 5:00	-7.0	-9.5	82	4	4.0	100.19	Snow
146	1/7/2012 2:00	-8.1	-11.1	79	4	19.3	100.15	Cloudy
...	...	...	...	...	...	...	...	...
8768	12/31/2012 8:00	-8.6	-10.3	87	4	3.2	101.14	Snow Showers
8769	12/31/2012 9:00	-8.1	-9.6	89	4	2.4	101.09	Snow
8770	12/31/2012 10:00	-7.4	-8.9	89	4	6.4	101.05	Snow,Fog
8772	12/31/2012 12:00	-5.8	-7.5	88	4	12.9	100.78	Snow
8773	12/31/2012 13:00	-4.6	-6.6	86	4	12.9	100.63	Snow

474 rows × 8 columns

```
In [26]: # find Null values
         df.isnull().sum()
Out[26]: Date/Time           0
         Temp_C              0
         Dew Point Temp_C    0
```

Fig2.6 Showing relationship between variables

To generate descriptive statistics for the DataFrame and then transpose the result to make it more readable. The describe () method provides summary statistics for the numerical columns in the DataFrame, such as count, mean, standard deviation, minimum, 25th percentile, median (50th percentile), 75th percentile, and max.

```
In [29]: df.describe().transpose()
Out[29]:
```

	count	mean	std	min	25%	50%	75%	max
Temp_C	8784.0	8.798144	11.687883	-23.30	0.10	9.30	18.80	33.00
Dew Point Temp_C	8784.0	2.555294	10.883072	-28.50	-5.90	3.30	11.80	24.40
Humidity	8784.0	67.431694	16.918881	18.00	56.00	68.00	81.00	100.00
Wind Speed_km/h	8784.0	14.945469	8.688696	0.00	9.00	13.00	20.00	83.00
Visibility_km	8784.0	27.664447	12.622688	0.20	24.10	25.00	25.00	48.30
Press_kPa	8784.0	101.051623	0.844005	97.52	100.56	101.07	101.59	103.65

Fig2.7 Showing deeper understanding of the underlying trends

After Exploratory Data Analysis (EDA) the data is ready for any form of visualization but we must first display all the requirements so as the code can fetch for visualization, below is some of the displays.

2.4 Display all records where the weather is snow

```
In [30]: df[df['Weather condition'] == 'Snow']
Out[30]:
```

	Date/Time	Temp_C	Dew Point Temp_C	Humidity	Wind Speed_km/h	Visibility_km	Press_kPa	Weather condition
55	1/3/2012 7:00	-14.0	-19.5	63	19	25.0	100.95	Snow
84	1/4/2012 12:00	-13.7	-21.7	51	11	24.1	101.25	Snow
86	1/4/2012 14:00	-11.3	-19.0	53	7	19.3	100.97	Snow
87	1/4/2012 15:00	-10.2	-16.3	61	11	9.7	100.89	Snow
88	1/4/2012 16:00	-9.4	-15.5	61	13	19.3	100.79	Snow
...	...	...	...	...	...	...	...	...
8779	12/31/2012 19:00	0.1	-2.7	81	30	9.7	100.13	Snow
8780	12/31/2012 20:00	0.2	-2.4	83	24	9.7	100.03	Snow
8781	12/31/2012 21:00	-0.5	-1.5	93	28	4.8	99.95	Snow
8782	12/31/2012 22:00	-0.2	-1.8	89	28	9.7	99.91	Snow
8783	12/31/2012 23:00	0.0	-2.1	86	30	11.3	99.89	Snow

390 rows × 8 columns

```
In [31]: df[df['Weather condition'].str.contains('Snow')].head(50)
Out[31]:
```

	Date/Time	Temp_C	Dew Point Temp_C	Humidity	Wind Speed_km/h	Visibility_km	Press_kPa	Weather condition
41	1/2/2012 17:00	-2.1	-9.5	57	22	25.0	99.66	Snow Showers
44	1/2/2012 20:00	-5.6	-13.4	54	24	25.0	100.07	Snow Showers

2.5 Display all records where wind speed reached 12 km/h and a 25 km visibility

```
In [33]: df[(df['Wind Speed_km/h'] > 12) & (df['Visibility_km'] == 25)].head(10)
```

Out[33]:

	Date/Time	Temp_C	Dew Point Temp_C	Humidity	Wind Speed_km/h	Visibility_km	Press_kPa	Weather condition
20	1/1/2012 20:00	3.2	1.3	87	19	25.0	99.50	Cloudy
21	1/1/2012 21:00	4.0	1.7	85	20	25.0	99.39	Cloudy
23	1/1/2012 23:00	5.3	2.0	79	30	25.0	99.31	Cloudy
24	1/2/2012 0:00	5.2	1.5	77	35	25.0	99.26	Rain Showers
25	1/2/2012 1:00	4.6	0.0	72	39	25.0	99.26	Cloudy
26	1/2/2012 2:00	3.9	-0.9	71	32	25.0	99.26	Mostly Cloudy
27	1/2/2012 3:00	3.7	-1.5	69	33	25.0	99.30	Mostly Cloudy
28	1/2/2012 4:00	2.9	-2.3	69	32	25.0	99.26	Mostly Cloudy
29	1/2/2012 5:00	2.6	-2.3	70	32	25.0	99.21	Mostly Cloudy
30	1/2/2012 6:00	2.3	-2.6	70	26	25.0	99.18	Mostly Cloudy

group by ('Weather condition'): This method is used to group the data based on the unique values in the 'Weather condition' column. It creates separate groups for each unique value in the specified column.

```
In [34]: df.groupby('Weather condition').mean()
```

Out[34]:

Weather condition	Temp_C	Dew Point Temp_C	Humidity	Wind Speed_km/h	Visibility_km	Press_kPa
Clear	6.825716	0.089367	64.497738	10.557315	30.153243	101.587443
Cloudy	7.970544	2.375810	69.592593	16.127315	26.625752	100.911441
Drizzle	7.353659	5.504878	88.243902	16.097561	17.931707	100.435366
Drizzle,Fog	8.067500	7.033750	93.275000	11.862500	5.257500	100.786625
Drizzle,Ice Pellets,Fog	0.400000	-0.700000	92.000000	20.000000	4.000000	100.790000
Drizzle,Snow	1.050000	0.150000	93.500000	14.000000	10.500000	100.890000
Drizzle,Snow,Fog	0.693333	0.120000	95.866667	15.533333	5.513333	99.281333
Fog	4.303333	3.159333	92.286667	7.946667	6.248000	101.184067
Freezing Drizzle	-5.657143	-8.000000	83.571429	16.571429	9.200000	100.202857
Freezing Drizzle,Fog	-2.533333	-4.183333	88.500000	17.000000	5.266667	100.441667
Freezing Drizzle,Haze	-5.433333	-8.000000	82.000000	10.333333	2.666667	100.316667
Freezing Drizzle,Snow	-5.109091	-7.072727	86.090909	16.272727	5.872727	100.520909
Freezing Fog	-7.575000	-9.250000	87.750000	4.750000	0.650000	102.320000
Freezing Rain	-3.885714	-6.078571	84.642857	19.214286	8.242857	99.647143
Freezing Rain,Fog	-2.225000	-3.750000	89.500000	15.500000	7.550000	99.945000
Freezing Rain,Haze	-4.900000	-7.450000	82.500000	7.500000	2.400000	100.375000
Freezing Rain,Ice Pellets,Fog	-2.600000	-3.700000	92.000000	28.000000	8.000000	100.950000
Freezing Rain,Snow Grains	-5.000000	-7.300000	84.000000	32.000000	4.800000	98.560000

Fig3.0 Snippet displaying groupby function

2.6 Display all records where the weather is foggy.

In [36]:
```
df[df['Weather condition'] == 'Fog'].head(10)
```

Out[36]:

	Date/Time	Temp_C	Dew Point Temp_C	Humidity	Wind Speed_km/h	Visibility_km	Press_kPa	Weather condition
0	1/1/2012 0:00	-1.8	-3.9	86	4	8.0	101.24	Fog
1	1/1/2012 1:00	-1.8	-3.7	87	4	8.0	101.24	Fog
4	1/1/2012 4:00	-1.5	-3.3	88	7	4.8	101.23	Fog
5	1/1/2012 5:00	-1.4	-3.3	87	9	6.4	101.27	Fog
6	1/1/2012 6:00	-1.5	-3.1	89	7	6.4	101.29	Fog
7	1/1/2012 7:00	-1.4	-3.6	85	7	8.0	101.26	Fog
8	1/1/2012 8:00	-1.4	-3.6	85	9	8.0	101.23	Fog
9	1/1/2012 9:00	-1.3	-3.1	88	15	4.0	101.20	Fog
10	1/1/2012 10:00	-1.0	-2.3	91	9	1.2	101.15	Fog
11	1/1/2012 11:00	-0.5	-2.1	89	7	4.0	100.98	Fog

2.7 Display all records where the weather is Clear.

In [37]:
```
df[(df['Weather condition'] == 'Clear') | (df['Visibility_km'] > 20)]
```

Out[37]:

	Date/Time	Temp_C	Dew Point Temp_C	Humidity	Wind Speed_km/h	Visibility_km	Press_kPa	Weather condition
20	1/1/2012 20:00	3.2	1.3	87	19	25.0	99.50	Cloudy
21	1/1/2012 21:00	4.0	1.7	85	20	25.0	99.39	Cloudy
23	1/1/2012 23:00	5.3	2.0	79	30	25.0	99.31	Cloudy
24	1/2/2012 0:00	5.2	1.5	77	35	25.0	99.26	Rain Showers
25	1/2/2012 1:00	4.6	0.0	72	39	25.0	99.26	Cloudy
...	...	...	...	...	...	...	...	...
8762	12/31/2012 2:00	-10.1	-13.4	77	9	25.0	101.45	Cloudy
8763	12/31/2012 3:00	-11.8	-14.4	81	6	25.0	101.42	Mostly Cloudy
8764	12/31/2012 4:00	-10.5	-12.8	83	11	25.0	101.34	Cloudy
8765	12/31/2012 5:00	-10.2	-12.4	84	6	25.0	101.28	Cloudy
8766	12/31/2012 6:00	-9.7	-11.7	85	4	25.0	101.23	Cloudy

7302 rows × 8 columns

2.8 Display all records where the weather is Clear and Humidity is greater than 50.

In [38]:
```
# Weather is Clear and 'Relative humidity' is Greater than 50
#visibility is above 40
df[(df['Weather condition'] == 'Clear') & (df['Humidity'] > 40) | (df['Visibility_km'] >30)].head(20)
```

Out[38]:

	Date/Time	Temp_C	Dew Point Temp_C	Humidity	Wind Speed_km/h	Visibility_km	Press_kPa	Weather condition
67	1/3/2012 19:00	-16.9	-24.8	50	24	25.0	101.74	Clear
106	1/5/2012 10:00	-6.0	-10.0	73	17	48.3	100.45	Mainly Clear
107	1/5/2012 11:00	-5.6	-10.2	70	22	48.3	100.41	Mainly Clear
108	1/5/2012 12:00	-4.7	-9.6	69	20	48.3	100.38	Mainly Clear
109	1/5/2012 13:00	-4.4	-9.7	66	26	48.3	100.40	Mainly Clear
110	1/5/2012 14:00	-5.1	-10.7	65	22	48.3	100.46	Mainly Clear
111	1/5/2012 15:00	-4.3	-12.0	55	26	48.3	100.52	Mainly Clear
114	1/5/2012 18:00	-7.1	-14.4	56	11	25.0	100.71	Clear
115	1/5/2012 19:00	-9.2	-15.4	61	7	25.0	100.80	Clear
116	1/5/2012 20:00	-9.8	-15.7	62	9	25.0	100.83	Clear
117	1/5/2012 21:00	-9.0	-14.8	63	13	25.0	100.83	Clear
183	1/8/2012 15:00	-6.6	-12.9	61	20	48.3	102.04	Mostly Cloudy
241	1/11/2012 1:00	-10.7	-17.8	56	17	25.0	101.49	Clear
242	1/11/2012 2:00	-12.0	-18.9	56	19	25.0	101.57	Clear
243	1/11/2012 3:00	-12.7	-19.4	57	19	25.0	101.64	Clear
244	1/11/2012 4:00	-13.4	-20.1	57	17	25.0	101.66	Clear
324	1/14/2012 12:00	-17.5	-23.8	58	20	48.3	101.16	Mostly Cloudy
325	1/14/2012 13:00	-17.1	-24.1	55	17	48.3	101.18	Mainly Clear

column = df.columns: This line stores the column names of the DataFrame df into the variable column. The df. columns attribute returns a pandas Index object containing all the column names of the DataFrame,

for columns in column: This line starts a loop where the variable columns iterate through each column name in the Index object column.

print("->", columns): This line prints each column name, preceded by the arrow symbol '->'.

```
In [39]:  column = df.columns
          for columns in column:
              print("-
                  >",columns)

-> Date/Time
-> Temp_C
-> Dew Point Temp_C
-> Humidity
-> Wind Speed_km/h
-> Visibility_km
-> Press_kPa
-> Weather condition
```

CHAPTER III

VISUALIZATION TECHNIQUES

3 Data visualization

Data visualization is a powerful technique used to present information in a graphical format, making complex datasets more accessible and understandable. It plays a crucial role in data analysis, enabling data scientists, analysts, and decision-makers to derive insights, identify patterns, and communicate findings effectively.

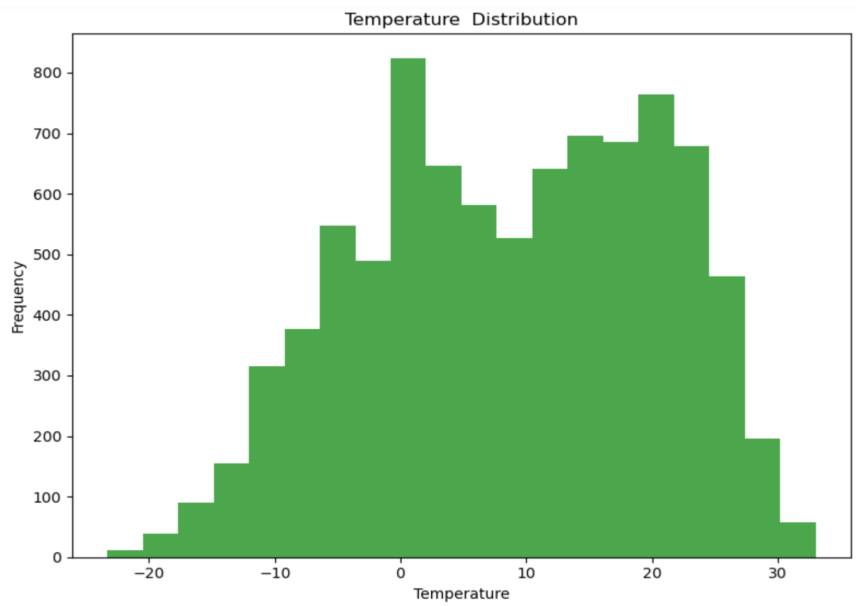
Will make use of histogram, heat map, scatterplot, boxplot and count plot

3.1 Histogram

A histogram is a diagram that shows how numerical data are distributed. It is composed of vertical bars, each of which represents a bin of values, and whose height denotes the frequency or number of data points that fall within that bin.

Implementing this in weather data

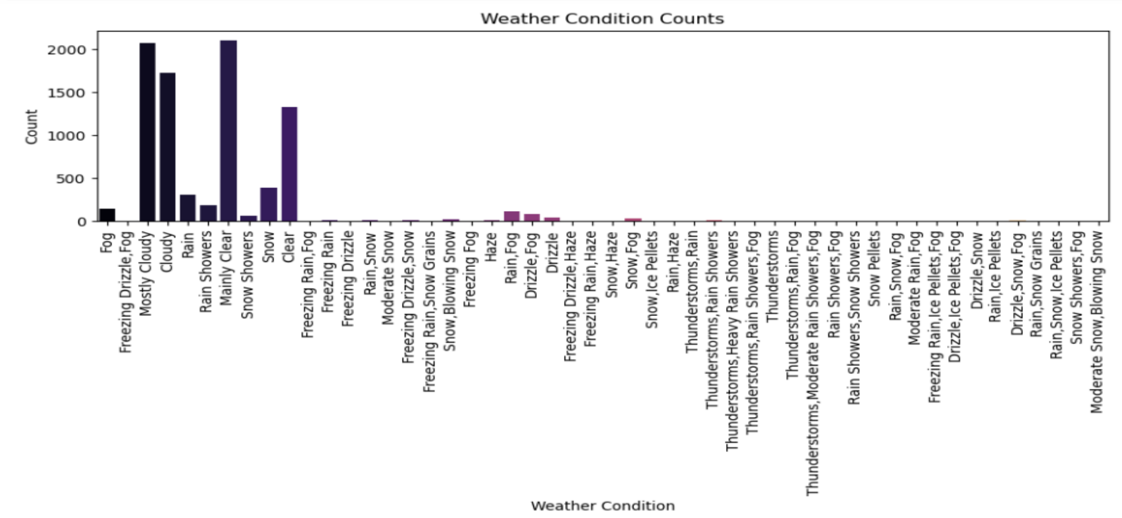
```
In [54]:
plt.figure(figsize=(8, 6))
plt.hist(df['Temp_C'], bins=20, color='green', alpha=0.7)
plt.xlabel('Temperature')
plt.ylabel('Frequency')
plt.title('Temperature  Distribution')
plt.tight_layout()
plt.show()
```



3.2 Count plot

A count plot is a type of bar plot used to visualize the frequency of categorical data. It displays the count of unique values in a categorical variable as bars, with each bar representing a category and its height indicating the count. As shown below is weather count plot.

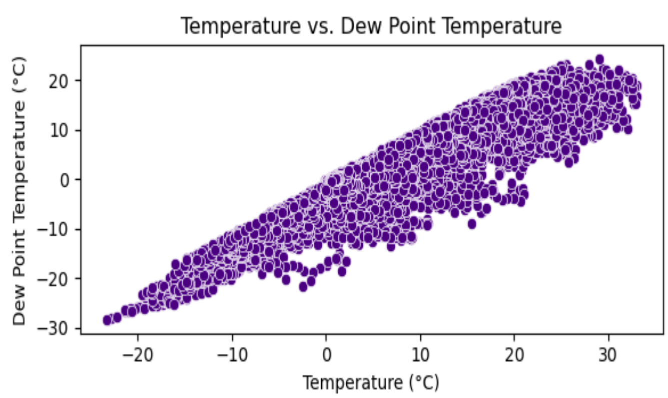
```
In [14]:
sns.countplot(x=df['Weather'] ,data=df, palette='magma')
plt.xlabel('Weather Condition')
plt.ylabel('Count')
plt.title('Weather Condition Counts')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



3.3 Scatter Plot

A scatter plot is a two-dimensional graph with dots on it to represent individual data points. It is used to show how two numerical variables relate to one another. Each dot in the scatter plot represents a data point, and the values of the two variables decide where on the graph each dot is located. This is displayed below

In [20]:
```
plt.figure(figsize=(6, 3))
sns.scatterplot(x='Temp_C', y='Dew Point Temp_C', data=df, color='indigo')
plt.xlabel('Temperature (°C)')
plt.ylabel('Dew Point Temperature (°C)')
plt.title('Temperature vs. Dew Point Temperature')
plt.tight_layout()
plt.show()
```

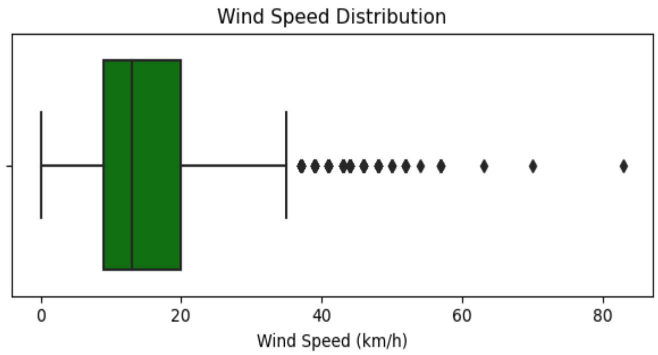


3.3 Boxplot

A boxplot, commonly referred to as a box-and-whisker plot, is a common visual representation of how numerical data are distributed.

The whiskers extend to the minimum and highest values within a certain range, and the box in the plot denotes the interquartile range (IQR). Boxplots can be used to compare data distributions, find outliers, and gain understanding of the variability and spread of the data.

```
In [21]:
plt.figure(figsize=(6, 3))
sns.boxplot(x=df['Wind Speed_km/h'], color='green')
plt.xlabel('Wind Speed (km/h)')
plt.title('Wind Speed Distribution')
plt.tight_layout()
plt.show()
```



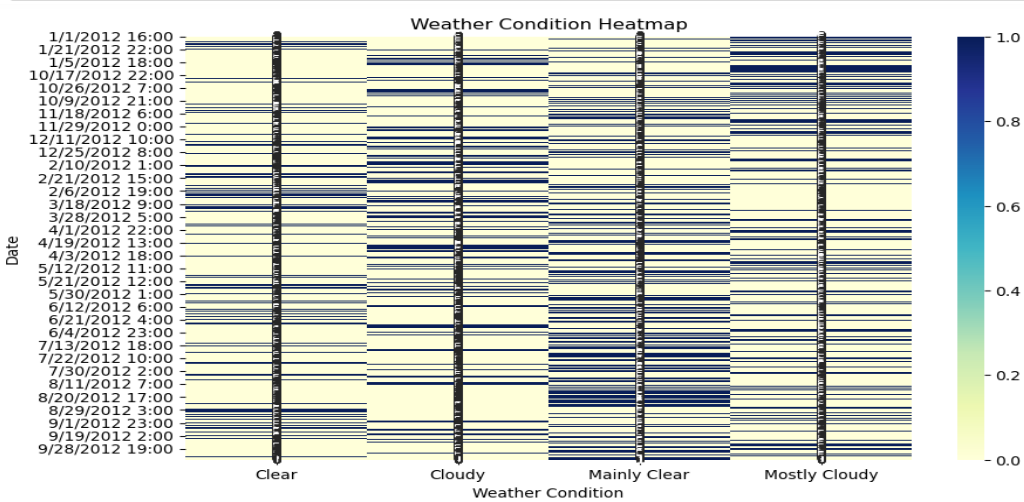
3.4 Heat Map

Individual values in a matrix are depicted as colors in a heatmap, which is a graphical representation of data. Each cell in the matrix has a particular color depending on its value, and the data values are encoded using a color scale. Heatmap are frequently used to show trends in large datasets, visualize correlation matrices, and give a visual overview of data relationships. They are useful for

```
In [22]:
# Filter the DataFrame to include only the specified weather categories
allowed_weather = ['Clear', 'Cloudy', 'Mainly Clear', 'Mostly Cloudy']
df_filtered = df[df['Weather'].isin(allowed_weather)]

# Create a contingency table
contingency_table = pd.crosstab(index=df_filtered['Date/Time'], columns=df_filtered['Weather'])

# Create the heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(contingency_table, cmap='YlGnBu', cbar=True, annot=True, fmt='d')
plt.title('Weather Condition Heatmap')
plt.xlabel('Weather Condition')
plt.ylabel('Date')
plt.show()
```



CHAPTER IV

RESULTS

4.1 Summary of data analysis findings

The weather dataset shows that there is a positive correlation between the various variables contained in the data, according to the data analysis that has been done and the visual presentation that has been created and is suitable for presentation and machine learning applications. As a result, the weather is ideal for agricultural practice with a range of (-20-30°C), which benefits the enterprises, but it is difficult for people living there because it is frequently cold and windy.

4.2 Discussion

I can say that the necessary goals have been met after performing the analysis on the Weather dataset. The data has first been cleaned to remove any unnecessary information, and then appropriate information has been added to the current data for quick and easy analysis.

4.2.1 Limitations and assumptions

Among the limitations encountered throughout the analysis project are;

- Having to erase some variables for accurate result.
- Having to convert data to other form so as it can run in the anaconda package.
- Hanging and malfunctioning of device.

A project's hypothesis is a key element. Several of the presumptions were;

- Any data can be analyzed.
- One can acquire data from any organization.

4.3 Conclusion and Recommendation

The dataset has enormous potential for a wide range of applications, such as climate research, weather forecasting, disaster management, and the development of renewable energy sources. This helpful tool can be used by researchers, meteorologists, data scientists, and climate enthusiasts to obtain a deeper understanding of weather occurrences and make wise judgments.

Some of the recommendation:

We advise making this weather dataset publicly accessible via an authorized data repository to optimize its usefulness and impact. Initiatives involving open data promote cooperation and motivate scholars from all around the world to examine, verify, and expand the dataset.

4.4 References

Niranjanamurthy. M. Hemant. (2012). *Advances In Data Science and Analytics*.

The link to dataset:

https://drive.google.com/file/d/1Fc8hbicZTl_7JRVkrfjfyMu73w2-d127/view?usp=drive_link

The link to the above module:

https://drive.google.com/file/d/1DlHnW7kAm2iPjaUb9sbDbmy3GCDzoauI/view?usp=drive_link

The link to National Centers for Environmental Information (NCEI):

https://www.ncei.noaa.gov/

Here is the link to access State sale comparison portfolio:

https://public.tableau.com/views/State_Comparison_of_sales/comparisonofstatesales?:language=en-US&:display_count=n&:origin=viz_share_link